# Mukto Pakhi — Prototype (MVP) Blueprint

A practical, staged plan to launch a child-safety reporting network using a mobile app + web portal backed by a secure cloud database. Designed for rapid MVP delivery with a clear path to scale (AI, wearables) later.

| ## 1) Vision & Goals **Vision:** A trusted, privacy-first channel for communities to quickly report, triage, and respond to risks to children (abuse, trafficking, missing children), enabling faster action by guardians, authorities, and NGOs. |
| --- |
| ## 2) Personas & Core Use Cases **Personas:** - **Parent/Guardian (PG):** Reports concerns, receives alerts. - **Community Reporter (CR):** Submits tips/observations (optionally anonymous). - **NGO Caseworker (CW):** Reviews cases, coordinates response. - **Authority Liaison (AL):** Police/child services contact for escalations. - **Admin (AD):** Configures orgs, permissions, triage rules. |
| **Key flows:** 1. **Report a case (PG/CR):** Open app → “Report” → Fill short form / send chat message → Attach optional photo/audio → Auto-capture coarse GPS → Submit. 2. **Triage (CW):** Case queue → View details & location → Assign priority → Notify stakeholders. 3. **Alerts:** Rules engine triggers Email/SMS/Push to guardians, NGO teams, or authority liaisons. 4. **Follow-up:** Chat thread per case → status updates → resolution. |

## 3) MVP Scope & Non-Goals

**In-scope (MVP):** - Mobile app (React Native) with form + chat; Bengali + English. - Web portal (Next.js) for case management and maps. - Backend REST API (Node/NestJS or FastAPI) with JWT/OAuth. - PostgreSQL (+ PostGIS) for cases & geodata. - File storage (S3/GCS) for media with signed URLs. - SMS/Email via Twilio/SendGrid; Push via FCM/APNS. - WhatsApp Business API (via Twilio) for chat bridge. - Basic rules engine for alerts (thresholds, geofences, keywords).

**Out-of-scope (Phase 1):** - Wearables, on-device ML, advanced NLP moderation, offline maps.

| ## 4) High-Level Architecture ┌──────────────────┐ HTTPS ┌──────────────────────┐ │ Mobile App │ <─────────> │ API Gateway │ │ (React Native) │ │ (NestJS/FastAPI) │ └──────────────────┘ ├──────────────────────┤ ┌──────────────────┐ Web │ Auth (JWT/OAuth) │ │ Web Portal │ <────────> │ Case/Chat Services │ │ (Next.js) │ │ Alerts Engine │ └──────────────────┘ │ Media Proxy │ └─────────┬────────────┘ Pub/Sub events │ ┌────────▼──────────┐ │ PostgreSQL+GIS │ │ (RDS/CloudSQL) │ └────────┬──────────┘ Files (S3/GCS, signed URLs) │ ┌────────▼──────────┐ │ Object Storage │ └────────┬──────────┘ WhatsApp/Twilio/SendGrid/APNS/FCM │ ┌────────▼──────────┐ │ Notifications │ └───────────────────┘ |
| --- |
| ## 5) Data Model (MVP) **users**(id, phone/email, name, role [PG/CR/CW/AL/AD], org\_id, locale, status, created\_at) |
| **orgs**(id, name, type [NGO/AUTHORITY], contact\_email, region\_code) |
| **cases**(id, reporter\_id, org\_id, title, description, status [new/in\_review/escalated/resolved], priority, created\_at) |
| **locations**(id, case\_id, geom GEOGRAPHY(Point,4326), accuracy\_m, captured\_at) |
| **messages**(id, case\_id, sender\_id/null, channel [app/whatsapp/sms], text, media\_url, created\_at) |
| **alerts**(id, case\_id, rule\_id, channel [email/sms/push], recipients[], sent\_at, status) |
| **rules**(id, org\_id, type [keyword/geofence/severity], config JSONB, active) |
| **consents**(id, user\_id, scope, granted\_at, revoked\_at) |
| **audit\_logs**(id, actor\_id, action, entity\_type, entity\_id, metadata, created\_at) |

## 6) API (Selected Endpoints)

**Auth:** - POST /auth/otp/request → {phone/email} - POST /auth/otp/verify → {code} → JWT

**Cases:** - POST /cases → create case {title, description, gps, media} - GET /cases?status=new&org\_id= → list - GET /cases/:id → details (+ messages, locations) - PATCH /cases/:id → {status, priority, assignee}

**Messages:** - POST /cases/:id/messages → {text|media} - Webhook: /integrations/whatsapp (Twilio) → upsert message

**Alerts:** - POST /alerts/test → dry-run rule - POST /rules / PATCH /rules/:id → manage org rules

**Media:** - POST /media/sign → {filename,mime} → {upload\_url, file\_url}

| ## 7) Alert Logic (MVP) - **Keyword triggers:** e.g., “abduction”, “threat”, “injury”. - **Geofence triggers:** Case location enters pre-defined risky zones (markets, transit hubs). - **Severity triggers:** Priority ≥ High. - **Recipient mapping:** By org & region (guardian(s), NGO duty roster, authority contact). - **Channels:** Email (SendGrid), SMS/WhatsApp (Twilio), Push (FCM/APNS). |
| --- |
| ## 8) Privacy, Safety & Compliance - **Consent-first:** Explicit scopes for location, media, messaging. - **PII minimization:** Collect only what’s needed; redact sensitive content for non-privileged viewers. - **Encryption:** TLS in transit; AES-256 at rest. Secrets in KMS/Parameter Store. - **Access control:** RBAC (role + org scoping), row-level security for cases. - **Auditability:** Every read/write logged. - **Child safety:** CSAM reporting protocols, moderation queue for media. - **Data residency:** Choose region (e.g., ap-south-1) and disclose. |

## 9) Deployment & DevOps

* **Cloud:** AWS (RDS Postgres+PostGIS, S3, Lambda/EC2), or GCP equivalents.
* **CI/CD:** GitHub Actions → staging → prod with infra as code (Terraform).
* **Observability:** CloudWatch/Stackdriver, OpenTelemetry traces; uptime & alert SLAs.
* **Backups:** Daily DB snapshots, 30-day retention; object versioning.

| ## 10) Phased Roadmap **Phase 0 (2–3 wks):** Clickable prototype; fake data; stakeholder review. |
| --- |
| ## 11) Low-Fidelity Wireframes (ASCII) **Mobile — Home** |

## 12) Prototype Code Snippets

### 12.1 React Native — Report Form (Expo)

// ReportScreen.tsx  
import React, { useState } from 'react';  
import { View, TextInput, Button, Switch, Text } from 'react-native';  
import \* as Location from 'expo-location';  
  
export default function ReportScreen({ navigation }) {  
 const [title, setTitle] = useState('');  
 const [description, setDescription] = useState('');  
 const [shareLoc, setShareLoc] = useState(true);  
  
 const submit = async () => {  
 let coords = null;  
 if (shareLoc) {  
 const { status } = await Location.requestForegroundPermissionsAsync();  
 if (status === 'granted') {  
 const loc = await Location.getCurrentPositionAsync({ accuracy: Location.Accuracy.Balanced });  
 coords = { lat: loc.coords.latitude, lng: loc.coords.longitude, acc: loc.coords.accuracy };  
 }  
 }  
 await fetch(process.env.EXPO\_PUBLIC\_API\_URL + '/cases', {  
 method: 'POST', headers: { 'Content-Type': 'application/json' },  
 body: JSON.stringify({ title, description, gps: coords })  
 });  
 navigation.replace('Confirmation');  
 };  
  
 return (  
 <View style={{ padding: 16 }}>  
 <TextInput placeholder="Title" value={title} onChangeText={setTitle} />  
 <TextInput placeholder="What happened?" value={description} onChangeText={setDescription} multiline />  
 <View style={{ flexDirection: 'row', alignItems: 'center', marginVertical: 8 }}>  
 <Text>Share location</Text>  
 <Switch value={shareLoc} onValueChange={setShareLoc} />  
 </View>  
 <Button title="Submit" onPress={submit} />  
 </View>  
 );  
}

### 12.2 Backend — Express/NestJS-style Endpoint

// POST /cases  
app.post('/cases', async (req, res) => {  
 const { title, description, gps } = req.body;  
 const result = await db.one(  
 `INSERT INTO cases (title, description, status, priority) VALUES ($1,$2,'new','medium') RETURNING id`,  
 [title, description]  
 );  
 if (gps) {  
 await db.none(  
 `INSERT INTO locations (case\_id, geom, accuracy\_m, captured\_at)  
 VALUES ($1, ST\_SetSRID(ST\_MakePoint($2,$3),4326), $4, NOW())`,  
 [result.id, gps.lng, gps.lat, gps.acc || null]  
 );  
 }  
 publishEvent('case.created', { id: result.id });  
 res.json({ id: result.id });  
});

### 12.3 SQL — Core Tables (PostgreSQL + PostGIS)

CREATE TABLE cases (  
 id BIGSERIAL PRIMARY KEY,  
 reporter\_id BIGINT NULL,  
 org\_id BIGINT NULL,  
 title TEXT NOT NULL,  
 description TEXT,  
 status TEXT NOT NULL DEFAULT 'new',  
 priority TEXT NOT NULL DEFAULT 'medium',  
 created\_at TIMESTAMPTZ NOT NULL DEFAULT NOW()  
);  
  
CREATE TABLE locations (  
 id BIGSERIAL PRIMARY KEY,  
 case\_id BIGINT REFERENCES cases(id) ON DELETE CASCADE,  
 geom GEOGRAPHY(Point,4326) NOT NULL,  
 accuracy\_m REAL,  
 captured\_at TIMESTAMPTZ NOT NULL DEFAULT NOW()  
);  
  
CREATE INDEX idx\_locations\_geom ON locations USING GIST(geom);

### 12.4 Alert Worker — Pseudocode

# on event: case.created or message.created  
if keyword\_match(case.description) or is\_high\_priority(case) or in\_geofence(case.location):  
 recipients = resolve\_recipients(case)  
 for r in recipients:  
 if r.pref == 'sms': twilio.sms(r.phone, template(case))  
 elif r.pref == 'email': sendgrid.email(r.email, subject(case), body(case))  
 elif r.pref == 'push': fcm.push(r.device\_token, payload(case))  
 log\_alerts(case.id, recipients)

| ## 13) Testing & QA - Unit tests for rules & geofence logic. - Integration tests for OTP, case creation, media upload (signed URLs). - E2E happy-path (Cypress/Detox): report → triage → alert → resolve. - Load test alerts fan-out at 100–1000 events/min. |
| --- |
| ## 14) Success Metrics (MVP) - Time-to-triage (P50/P95). - % cases with location captured. - Alert delivery success rate (per channel). - Case resolution time & stakeholder satisfaction. |

## 15) Next Steps (Practical)

1. Stand up repo + infra skeleton (Terraform, CI).
2. Build Expo prototype (Report, Chat stub).
3. Stand up API + Postgres + PostGIS.
4. Hook up Twilio/SendGrid sandbox; test alert flows.
5. Pilot with 1–2 NGOs; iterate on forms & consent UX.

*End of MVP blueprint. This document is designed for rapid iteration—leave comments where you want deeper detail or changes.*